

AD-A073 088

MITRE CORP BEDFORD MA

F/G 9/2

ERROR CORRECTION CODING WITH NMOS MICROPROCESSORS: VOLUME II. A--ETC(U)

MAY 79 J R HAMALAINEN, E N SK006

F19628-78-C-0001

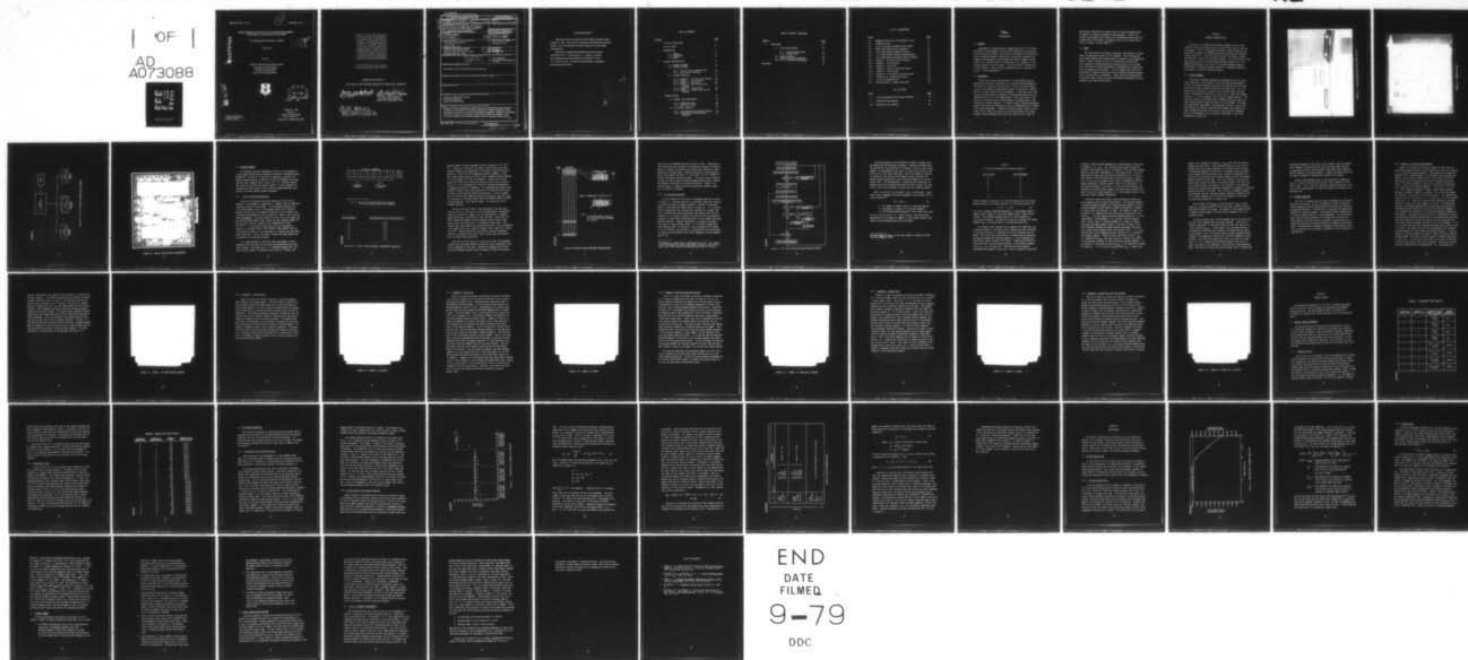
UNCLASSIFIED

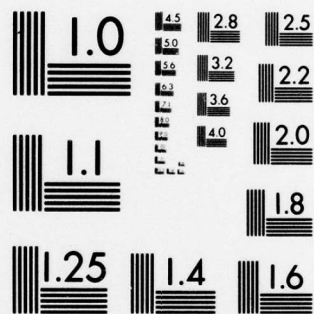
MTR-3618-VOL-2

ESD-TR-79-125-VOL-2

NL

| OF |
AD
A073088





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

ERROR CORRECTION CODING WITH NMOS MICROPROCESSORS:
A 6800-BASED 7,3 REED-SOLOMON DECODER

J.R. HAMALAINEN AND ERIC N. SKOOG

MAY 1979

LEVEL

Prepared for

DEPUTY FOR DEVELOPMENT PLANS

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts

DA 073088

DDC FILE COPY



DDC
AUG 24 1979
A

Approved for public release;
distribution unlimited.

Project No. 7010

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

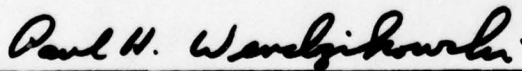
Contract No. F19628-78-C-0001

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

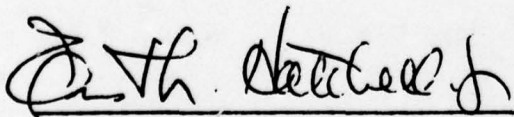
This technical report has been reviewed and is approved for publication.



PAUL H. WENDZIKOWSKI, Capt, USAF
Project Officer



WILLIAM M. SMITH, Jr., Col, USAF
Director, Technological and
Functional Area Planning
Deputy for Development Plans



ERNEST L. HATCHELL JR., Colonel, USAF
Assistant Deputy for Development Plans

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-79-125 - Vol II 2	2. GOVT ACCESSION NO. Volume II	3. RECIPIENT'S CATALOG NUMBER 9 Technical report
4. TITLE (and Subtitle) ERROR CORRECTION CODING WITH NMCS MICROPROCESSORS: A 6800-BASED 7, 3 REED-SOLOMON DECODER.		5. TYPE OF REPORT & PERIOD COVERED
7. AUTHOR(s) J. R. /Hamalainen Eric N. /Skoog		6. PERFORMING ORG. REPORT NUMBER MTR-3618-VOL-II 2
		8. CONTRACT OR GRANT NUMBER(s) F19628-78-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation P.O. Box 208 Bedford, MA 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project No. 7010
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Development Plans Electronic Systems Division, AFSC Hanscom AFB, MA 01731		12. REPORT DATE May 1979
		13. NUMBER OF PAGES 60
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 60p		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) A072 982		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) ERROR CORRECTION CODING MICROPROCESSORS REED-SOLOMON CODE		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Design, operation and testing of a table-look-up microprocessor-based (Motorola MC6800) (7,3) Reed-Solomon decoder is presented. Decoder operation is fully illustrated by the use of error and erasure pattern examples. Exhaustive (complete) and random (Monte Carlo) testing is employed to exercise the decoder. Test results are analyzed and conclusions drawn relative to decoder performance.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

235 050

Lm

ACKNOWLEDGMENT

This report has been prepared by The MITRE Corporation under Project No. 7010. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

The authors gratefully acknowledge the hardware support contributed by R. A. Gamache and J. E. Tolliver, as well as the excellent programming assistance provided by R. Drury, C. E. Pearson, and J. Terzakis during the design, construction, and testing of the decoder.

Accession For	
NTIS GRA&I	
DDC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist.	Avail and/or special
A	

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
LIST OF ILLUSTRATIONS	5
LIST OF TABLES	5
1 INTRODUCTION	6
1.1 PURPOSE	6
1.2 BACKGROUND	6
1.3 SCOPE	7
2 DECODER IMPLEMENTATION	8
2.1 DECODER HARDWARE	8
2.2 DECODER SOFTWARE	13
2.2.1 (7,3) R-S Code Representation	13
2.2.2 The Decoding Algorithm	17
2.3 DECODER OPERATION	23
2.3.1 Example 1 - No Errors and Erasures	24
2.3.2 Example 2 - Five Erasures	27
2.3.3 Example 3 - Two Errors	29
2.3.4 Example 4 - One Error and Two Erasures	31
2.3.5 Example 5 - Three Errors	33
2.3.6 Example 6 - Three Errors and Two Erasures	35
3 DECODER TESTING	37
3.1 DECODER TESTS AND RESULTS	37
3.1.1 Exhaustive Tests	37
3.1.2 Monte Carlo Tests	39
3.2 TEST RESULTS ANALYSIS	41
3.2.1 Correctable Error/Erasure Patterns	41
3.2.2 Uncorrectable Error/Erasure Patterns	42

TABLE OF CONTENTS (CONCLUDED)

<u>Section</u>	<u>Page</u>
4 CONCLUSIONS	49
4.1 SYSTEM IMPLICATIONS	49
4.1.1 Average Decoding Rate	49
4.1.2 Decoder Delay	52
4.2 LESSONS LEARNED	53
4.3 ERROR CODING SYSTEM TESTING	55
4.4 (7,3) R-S DECODER IMPROVEMENTS	56
REFERENCES	59

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
2-1 MC6800 Microprocessor Development Facility	9
2-2 MC6800 CPU Card	10
2-3 (7,3) R-S Decoder Hardware Block Diagram	11
2-4 MOSTEK 3870 One-Chip Microcomputer	12
2-5 (7,3) Reed-Solomon Codeword Structure	14
2-6 (7,3) Reed-Solomon Codeword Representation	16
2-7 (7,3) Reed-Solomon Decoding Program Flowchart	18
2-8 Example 1 - No Errors and No Erasures	26
2-9 Example 2 - Five Erasures	28
2-10 Example 3 - Two Errors	30
2-11 Example 4 - One Error and Two Erasures	32
2-12 Example 5 - Three Errors	34
2-13 Example 6 - Three Errors and Two Erasures	36
3-1 Exhaustive vs. Random Test Results	43
3-2 (7,3) R-S Standard Array	46
4-1 Decoding Rate vs. Symbol Error Rate	50

LIST OF TABLES

<u>Table</u>	<u>Page</u>
I (7,3) Correctable Error/Erasure Patterns	20
II Exhaustive Test Results	38
III Monte Carlo Test Results	40

SECTION 1

INTRODUCTION

1.1 PURPOSE

Improvements in Communications, Command and Control (C^3) System reliability and throughput can be achieved through the incorporation of error correction coding. Specifically, the control of transmission errors induced by channel characteristic variations, equipment performance inconsistencies, and man-made interference whether unintentional (e.g., machine noise) or intentional (e.g., jamming), is the primary objective of error coding. This paper details the design of a single forward - error correcting decoder based on a low-cost LSI circuit - the microprocessor.

1.2 BACKGROUND

Project 7010 (Low Cost Electronics) is investigating new LSI technologies and their suitability for increasing the performance of C^3 systems and lowering life-cycle costs. Error correction coding can be a very effective system design element in achieving increased levels of system reliability and throughput. In the past, error coding has not been regularly employed in system designs because of the complexity and cost of its implementation. With the advent of low-cost LSI circuits and fairly recent advances in coding theory, this situation is rapidly changing. In order to exploit these developments, Project 7010 has been investigating low-cost implementations of simple error coding techniques. Attention has been recently focused on non-binary cyclic codes, and specifically the Reed-Solomon (R-S) Codes. Studies already completed have shown that the use of R-S codes on

approximated communication channels leads to improved communication throughput and therefore enhanced system operation. Details concerning the conceptual work behind utilizing NMOS microprocessors for decoding short block codes, specifically the (7,3) R-S code, can be found in Volume I of this report.[1]

1.3 SCOPE

This paper details the design, operation, and testing of a table-search microprocessor-based (7,3) R-S decoder. The microprocessor host system hardware and software design is detailed in an Engineering Record. Decoder operations are fully illustrated including specific examples of decoding beyond the bounds of the (7,3) R-S code. Error and erasure handling performance is explained as it affects decoder throughput and performance. Lastly, decoder test results are presented for complete and exhaustive error pattern testing and Monte Carlo simulated error environments. These results are analyzed and conclusions drawn relative to three main decoder performance criteria: decoding rate, delay and buffering.

SECTION 2

DECODER IMPLEMENTATION

The actual (7,3) Reed-Solomon decoder hardware is but a small subset of the system hardware used to interactively exercise and test the decoder. The (7,3) Reed-Solomon decoder software executes as an application program on an MC6800 microprocessor-based host (support) system (Figure 2-1). The host system's electrical design, software, and operation is detailed in another document. This section details the basic design of the (7,3) Reed-Solomon decoder hardware (i.e., the MC6800 microprocessor card) and its associated decoding software. Additionally, interactive decoder operation is completely demonstrated through six specific decoding examples.

2.1 DECODER HARDWARE

The amount of hardware necessary to implement the (7,3) R-S decoder represents a small amount of the total host system hardware. In fact, the entire decoding algorithm is executed by only six chips on the host system's Central Processing Unit (CPU) card. This minimum chip set (indicated in Figure 2-2 by the chips with light gray labels) implementation consists of a CPU, clock, 1 RAM, and 3 EPROMs. Figure 2-3 is a block diagram of the (7,3) R-S decoder hardware. The code table occupies 1536 bytes (two 1K EPROMs), while the actual decoder program requires about 256 bytes of a third EPROM. The temporary RAM storage requirement is approximately 20 bytes. Due to the small amounts of program, table and data storage, this specific decoder implementation could utilize one of the recently introduced one-chip microcomputers such as Mostek's 3870 shown in the microphotograph of Figure 2-4.

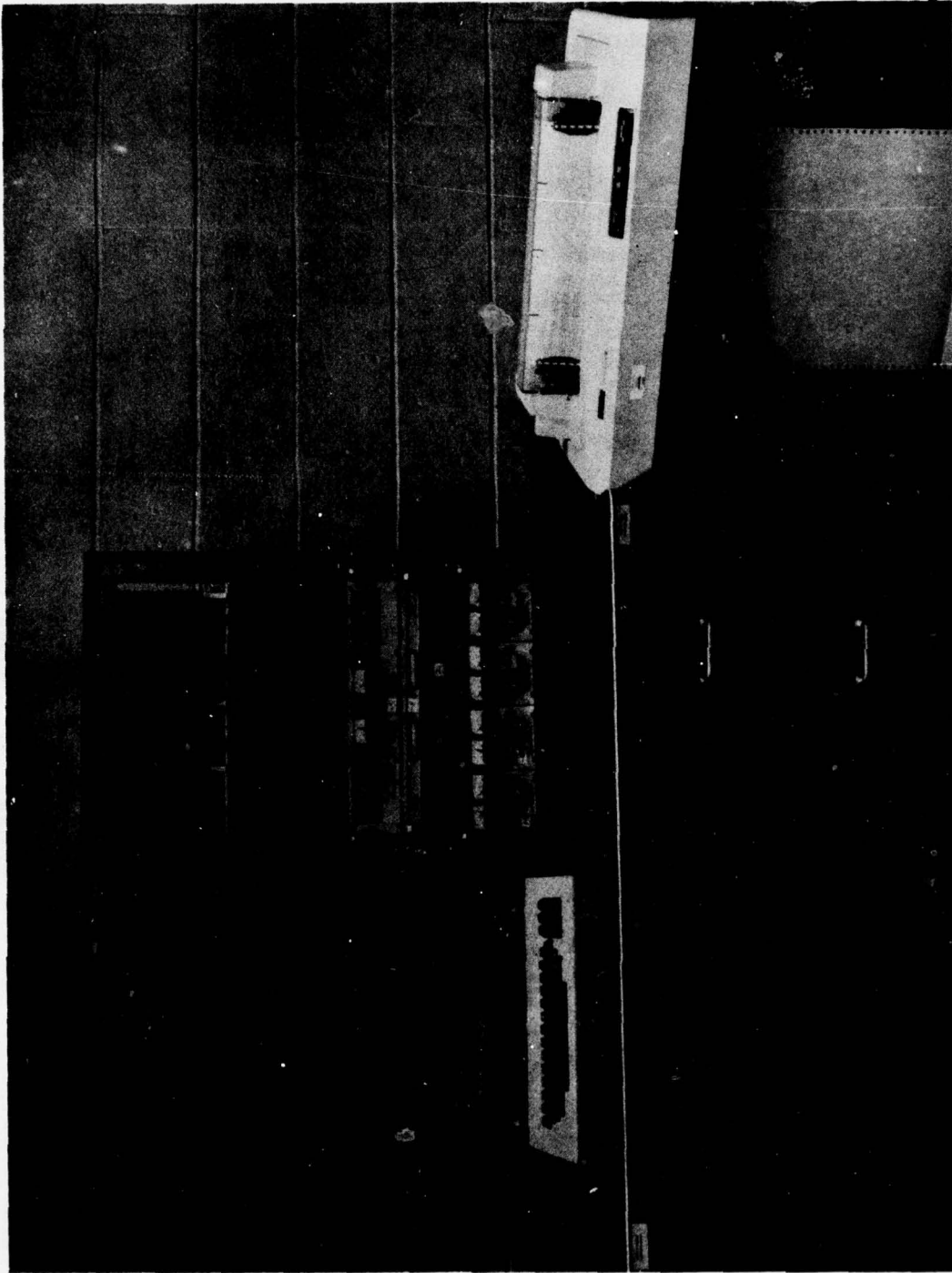


FIGURE 2-1: MC6800 MICROPROCESSOR DEVELOPMENT FACILITY

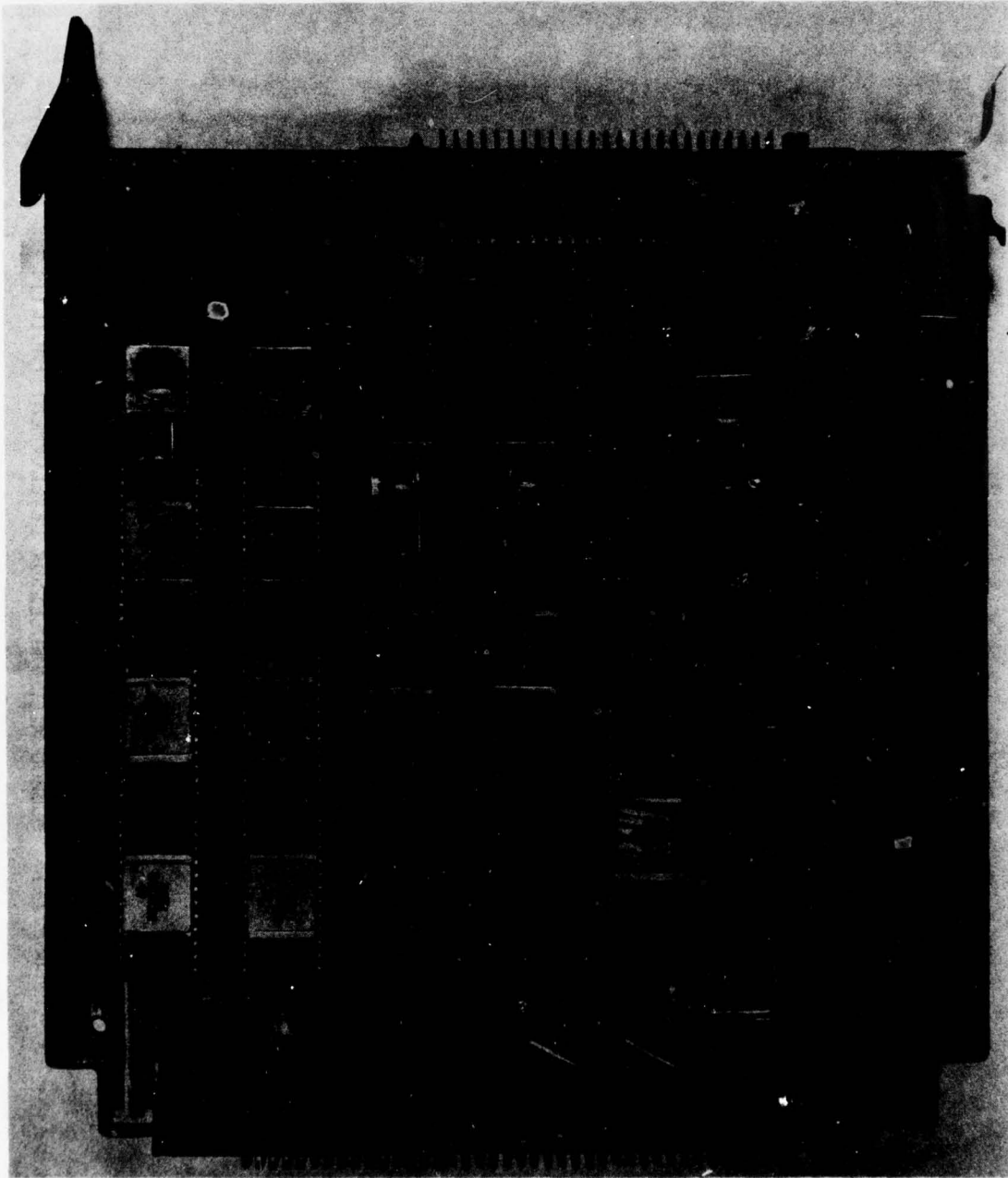


FIGURE 2-2: MC6800 CPU CARD

2A-62,648

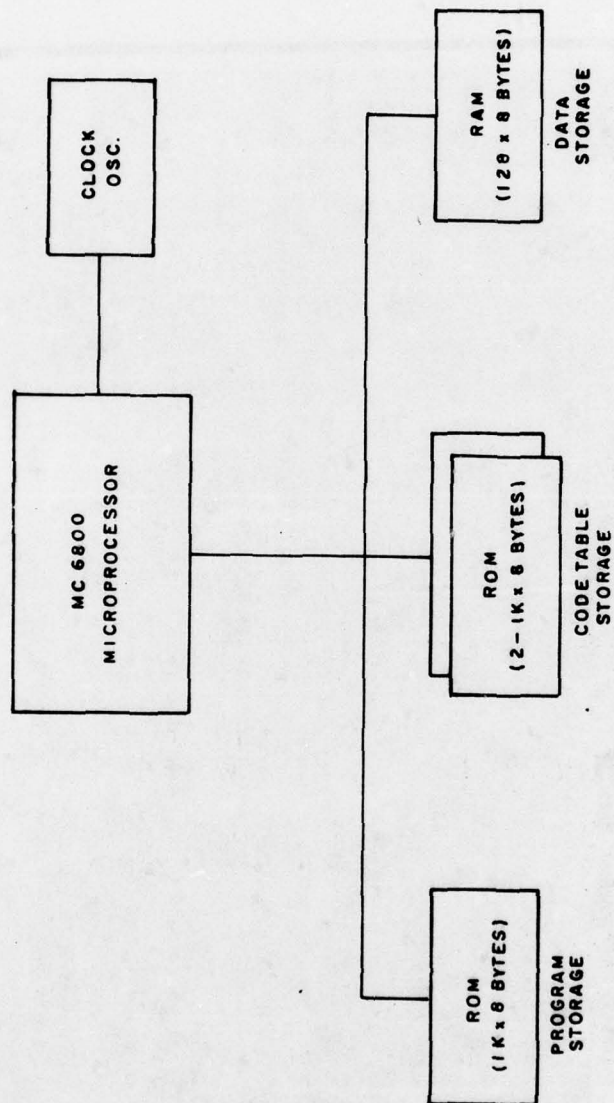
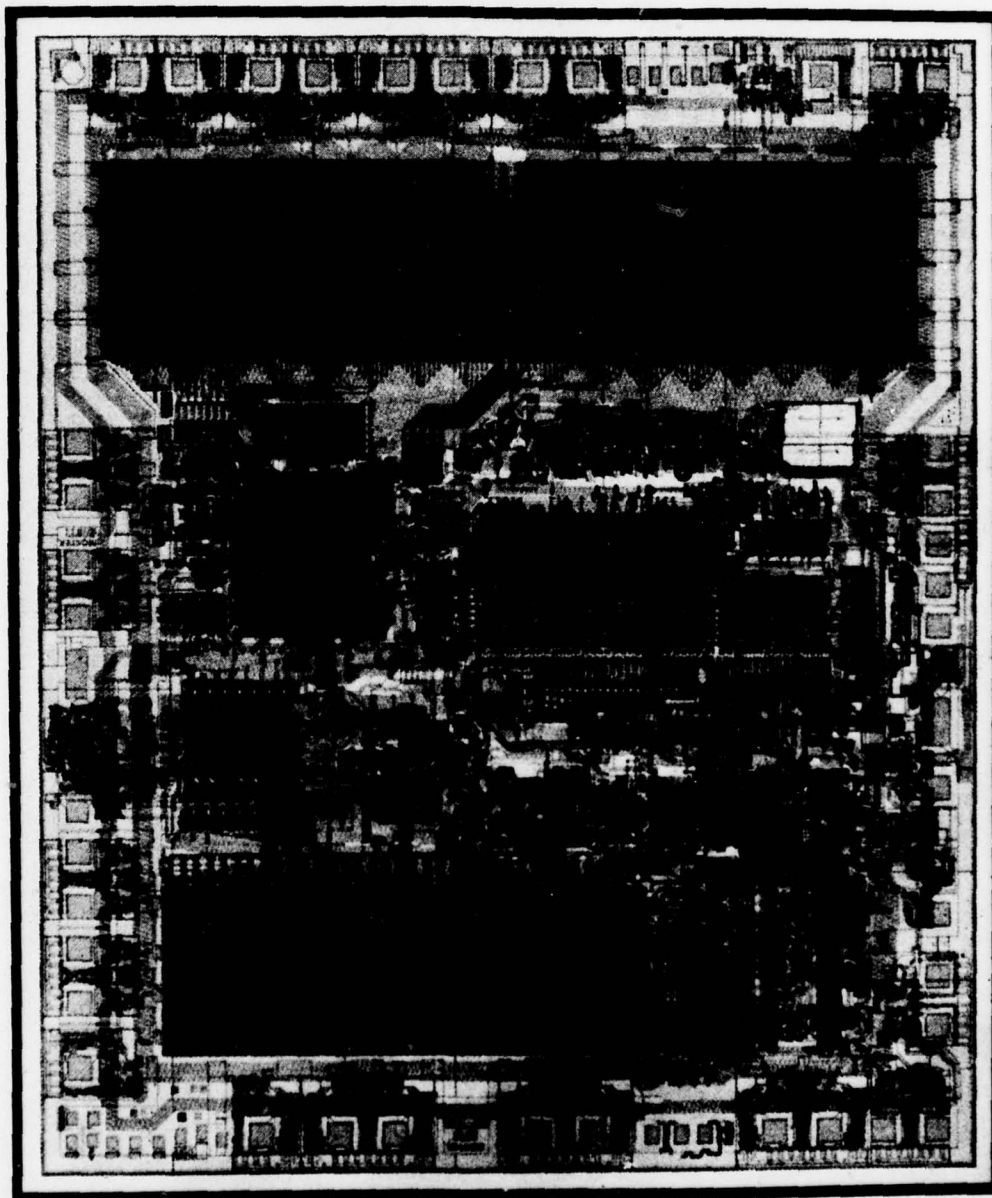


Figure 2-3 : (7,3) R-S DECODER HARDWARE BLOCK DIAGRAM



Mostek 3870.

FIGURE 2-4: MOSTEK 3870 ONE-CHIP MICROCOMPUTER

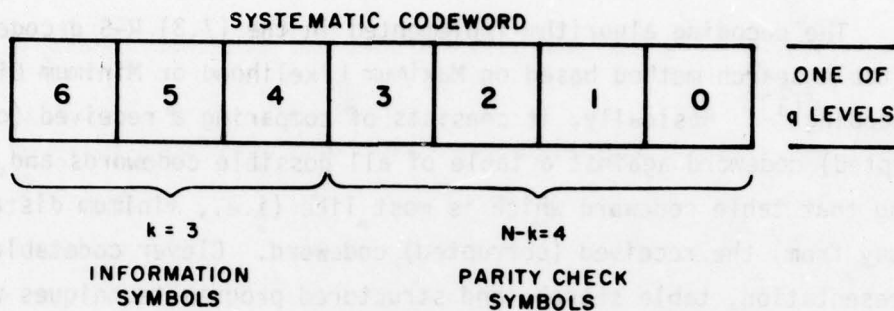
2.2 DECODER SOFTWARE

The decoding algorithm implemented in the (7,3) R-S decoder is a table search method based on Maximum Likelihood or Minimum Distance Decoding^[1]. Basically, it consists of comparing a received (corrupted) codeword against a table of all possible codewords and choosing that table codeword which is most like (i.e., minimum distance away from) the received (corrupted) codeword. Clever codetable representation, table search, and structured program techniques were applied to achieve maximum decoder throughput.

2.2.1 (7,3) R-S Code Representation

Figure 2-5 illustrates the block structure of the (7,3) R-S codeword. Each codeword is comprised of 7 symbols, each of which may represent any one of eight levels. In the encoding operation the first three symbols are set to correspond exactly to the desired message, and the last four symbols, called parity check symbols, are calculated and appended for error correction. Codes of this type are termed Systematic. The coding theory of linear cyclic BCH codes, and particularly Reed-Solomon codes, states that the codeword symbols of the (7,3) R-S code are elements of an extended Galois Field $GF(2^3)^{[1,2]}$. By picking an irreducible primitive polynomial of degree $m=3$, the eight finite field elements (code symbols) may, as illustrated in Figure 2-5, be represented as 3-tuples over the binary base field $GF(2)$. This is the manner in which the code symbols are represented in the decoding program.

It is then necessary to pack the seven code symbols of each codeword into 8-bit bytes for efficient data manipulation and codetable storage. Since there are $k = 3$ information symbols per codeword, and each symbol can represent any of the $q = 8$ levels, the



$$N = q - 1 = p^m - 1 = 7, \text{ \& } q = 8$$

$\therefore p^m = 2^3$ AND THE CODEWORD SYMBOLS ARE ELEMENTS OF $GF(2^3)$, A GALOIS FIELD OF 8 ELEMENTS

FIELD ELEMENTS

FIELD ELEMENTS AS 3-TUPLES OVER $GF(2)$

0	000
a^0	001
a^1	010
a^2	100
a^3	011
a^4	110
a^5	111
a^6	101

1A-52,849

Figure 2-5 : (7,3) REED SOLOMON CODEWORD STRUCTURE

maximum number of unique messages and hence codewords is $q^k = 8^3 = 512$. Figure 2-6 illustrates the manner in which the finite field element codewords are represented in the binary realm of the microprocessor. Each of the 512 codewords contains 7 symbols. Since each symbol can be any one of eight values, codeword storage requires 3 bits per symbol or 21 bits per codeword. These 21 bits can be packed into three 8-bit bytes with three bits left over as shown in Figure 2-6. The implementation scheme for codeword representation is highly microprocessor dependent. The representation scheme illustrated in Figure 2-6 has been optimized for maximum efficiency when utilized with the Motorola 6800 microprocessor. It will be noted that six of the seven codeword symbols are located in the same relative positions in the three bytes, while the seventh symbol occupies a unique position. This seventh symbol is (of necessity) split between two bytes.

The split has been coded to take advantage of the easy testability of bit #7, the sign bit of the microprocessor's accumulator register (a status bit common to all microprocessors). Codeword symbol #6-bit C_{20} is positioned in that location. Using the first three information symbols of the systematic code to form a table entry vector, the table search is initiated at that point. This assumes that these symbols were received correctly, a valid assumption in the majority of cases involving "decent" error rates (e.g., better than 10^{-3}). Maximizing correct table entry increases throughput because it minimizes table search time, in most cases resulting in only one table look-up operation.

Since the (7,3) R-S code is a valid cyclic code, the codetable storage could have been reduced by 85.55% by taking advantage of the cyclic properties of the code. Alternative programs were written using this method, but were found to adversely affect the decoding

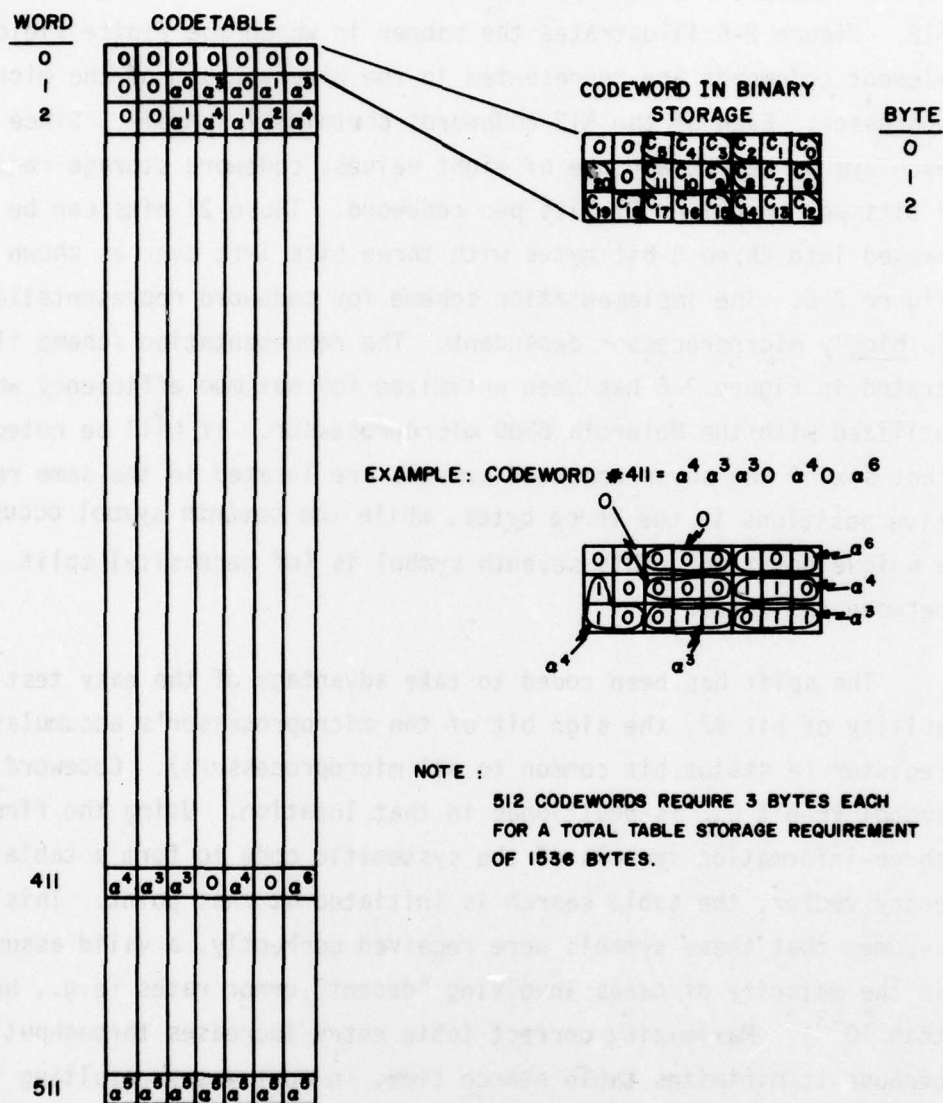


Figure 2-6: (7,3) REED-SOLOMON CODEWORD REPRESENTATION

rate due to the overhead required for cyclic shifts. Nevertheless, this alternative does represent an interesting tradeoff of decoding speed vs. storage requirements particularly when considering implementations of longer codes. Another alternative is the storage of only 4 code symbols instead of 7, since every (7,3) R-S codeword is uniquely defined by any three symbols. This would reduce codetable storage, but would require received codeword permutations to govern the compares and ensure error/erasure correction. However, the maximum number of compares would be dramatically reduced, making this an attractive tradeoff.

2.2.2 The Decoding Algorithm

A basic flowchart of the (7,3) R-S decoding algorithm is shown in Figure 2-7. Decoding operation is explained using this flowchart. A received (possibly corrupted) codeword is stored in scratchpad memory in the three byte format shown in Figure 2-6. It is assumed that the demodulator has converted the received m-ary symbols into their corresponding binary equivalents and transmits the received codeword (i.e., a block of three 8-bit bytes) to the microprocessor along with a fourth byte indicating which, if any, symbols have been declared erasures. Erasure^{*} declaration is assumed to include a "best guess" at the received symbol. It is further assumed that, since decoding time is a function of the corrupting error pattern, it is not constant, and therefore, a system buffer must be employed to store received codewords while other codewords are being decoded (see 4.1.2).

* An ERASURE is a symbol that is known to be in error. This simplifies the decoding task because the symbol location of the error is known, and it then remains to determine the value of the error.

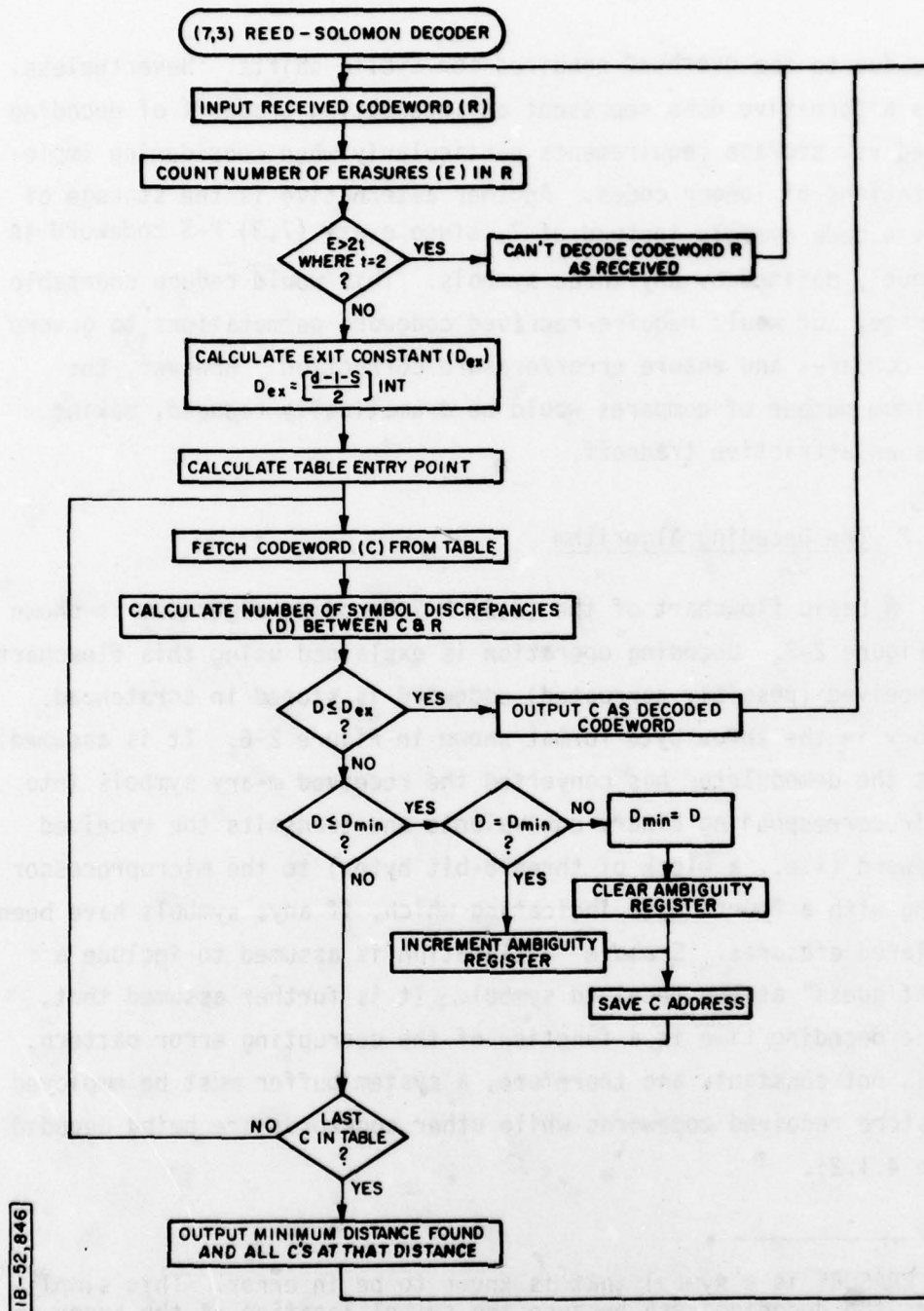


Figure 2-7 : (7,3) REED SOLOMON DECODING PROGRAM FLOWCHART

The microprocessor first evaluates the number of symbols that the demodulator has declared as erasures. Maximum likelihood decoding provides for the correction of a maximum of twice as many erased symbols, in the absence of any errors, as the error correction capability (t) of the linear block code. Should the number of erasures (E) exceed $2t$, then the received codeword cannot be decoded and further decoding operations on this received codeword are a waste of decoder time. The microprocessor, recognizing this event, does not attempt any further decoding operations and simply outputs the codeword as received and proceeds to decode the next received codeword.

Next, a so-called "Exit-Constant" (D_{ex}) is calculated. From a simple proof of Maximum Likelihood Decoding it can be shown that for correct decoding^[1]

$$2t + s \leq d - 1 \quad (1)$$

t = the number of symbol errors in a received codeword

s = the number of erasures in a received codeword

d = the minimum Hamming distance* of the linear block code

For the (7,3) R-S code $d = 5$. Shown in Table I are the possible combinations of the number of symbols in error and/or erased in any single received codeword that can be corrected.

*Minimum Hamming distance is the least number of symbols by which any two codewords differ.

Table I
(7,3) R-S Correctable Error/Erase Patterns

<u>No. of Errors</u>	<u>No. of Erasures</u>
0	1
0	2
0	3
0	4
1	0
1	1
1	2
2	0

Given the number of erasures in a received codeword and the minimum Hamming distance of the code, equation (1) can be manipulated into:

$$t \leq (d - 1 - s)/2 \quad (2)$$

For correct maximum likelihood decoding, t is the maximum number of symbol errors allowed in a received codeword given s known erased symbols. The truncated integer value of t will be defined as D_{ex} the exit constant. The utility of this exit constant will become evident later in the description of the decoding algorithm.

The codeword table provides valid codewords for comparison with a received codeword. The table search procedure has been structured to minimize the number of codewords that must be accessed and compared. This table search optimization is accomplished by beginning the search assuming the received codeword is a valid codeword and entering the table at exactly that entry. A table entry pointer is calculated from the three message symbols of the received codeword (bits $c_{12} - c_{20}$ in Figure 2-6). Note that with the codeword representation scheme of Figure 2-6, a one bit rotate-left operation

on byte 2 of the received codeword sets up the testing of three bits of symbol 6, i.e., $C_{18} - C_{20}$, since C_{18} and C_{20} are in the sign position of the bytes and C_{19} is in the carry. This efficient method results in the quick calculation of a table entry point. It may be noted that no rotate operation would be required at all if bit C_{18} were located in the most significant (sign) bit of byte 0. While this is true, that arrangement could result in abnormally difficult compare operations involving symbol 6 later on in the program. If the received codeword is correct, or if all errors and erasures are confined to the parity check symbols, the table entry procedure will result in only one fetch operation, i.e., the processor points to the described correct codeword immediately.

Using this table entry point, the microprocessor fetches one valid codeword from the table (three bytes). The selected codeword is compared symbol-by-symbol (not counting erased symbols) with the received codeword and the number of symbol discrepancies D (number of symbols that are different) is found. The number of symbol discrepancies D is compared with the previously calculated exist constant D_{ex} . If at any stage during the decoding operation, it is discovered that a valid codeword, i.e., an entry from the codeword table, differs from the received codeword by a number of symbols (not counting erased symbols) which is less than or equal to the exist constant, Maximum Likelihood Decoding guarantees that there are no other valid codewords which are any more likely. If $D \leq D_{ex}$, then the most likely codeword has been located and it is declared the decoded codeword. Should D not be less than or equal to D_{ex} , D is compared with D_{min} , the least value of D found to that point in the decoding procedure for that particular received codeword. If the latest D is less than or equal to D_{min} , it is specifically tested for equality. If D is found to exactly equal D_{min} , it means that

another valid codeword of distance $D = D_{\min}$ away from the received codeword has been found. This situation cannot exist for any correctable error and erasure patterns or for patterns which corrupt a transmitted codeword so badly that it is mapped closer to another valid codeword than the one which was originally transmitted (the weight of the corrupting error pattern exceeds one-half the minimum Hamming distance of the code). There are, however, error patterns whose weight exceeds the number of allowable symbol errors and also are an equal distance from several valid codewords. The algorithm detects such patterns and therefore represents an "attempt" at decoding beyond the bounds of the code. This is only an "attempt", because the ambiguity register indicates how many ambiguous codewords there are in these instances, and a list of the codewords is available as output (see example No. 5, Figure 2-12).

At this point if $D \neq D_{\min}$, it must be that $D < D_{\min}$ so that a new D_{\min} is declared equal to the D just found, and all previous ambiguity information is erased. The address of the codeword found is saved for later output if no more likely codeword(s) are subsequently found.

The next decision block in the flowchart asks if the entire 512 entries in the codeword table have been searched. (This point is reached immediately if D had been found to be not less than a previously declared D_{\min} .) If the answer is no, the table pointer is incremented and the next valid codeword is fetched from the table. If the answer is yes, the decoder outputs the best minimum distance (number of discrepancies) found and all the valid codewords at that distance. Note that this program exit point only occurs for codes with $t > 2$ (i.e., beyond the correctable bounds of the (7,3) R-S code), and results in an ambiguity in the possible transmitted codeword. Had the error pattern with weight greater than 2 corrupted the

transmitted codeword so badly that it was "mapped" closer to another valid codeword, the algorithm exit point would have been from the $D \leq D_{ex}$ decision block, and the decoder would never have recognized that it decoded to the wrong codeword. Indeed, the decoder sees this as simply an example of decoding an allowable error pattern.

The main time consuming loop in this algorithm is between the "Last C in table?" decision block and the "Fetch codeword (C) from table" block. This is the most vigorously optimized loop and is that part of the algorithm which is timed for purposes of estimating the decoding rate.

2.3 DECODER OPERATION

The (7,3) R-S decoder was specifically designed for interactive operation and testing. As such, an operator is able to choose any single codeword and error/erasure pattern and observe the results of the decoding operation. Alternatively, an operator can exhaustively generate all possible codeword - error/erasure pattern combinations or initiate random selection of these parameters. To illustrate decoder operation and performance, six examples will be explained. These examples have been carefully selected to show the results of decoding different error/erasure patterns on randomly selected codewords. The performance assessment parameters of the decoding operation are available via the decoder information fields displayed on a video monitor.

2.3.1 Example 1 - No Errors and Erasures

Figure 2-8 illustrates the monitor display for the case of a transmitted codeword that was not corrupted by any errors or erasures, i.e., a correctly received codeword. The program first prompts the operator for an "Input Message". The operator inputs three message symbols in their binary representation, in this case 100 100 101. The program uses this message to calculate a codeword table entry point. The entire codeword is read out of the table and displayed on the screen as the "Encoded Codeword": 100 100 101 111 101 110 111. Thus, codeword encoding is simply a table lookup procedure whereby a systematic codeword is produced. The system then prompts the operator to "Input Error and Erasure Pattern". The operator may indicate an error by typing in a "1", no error as a "0", or an erased symbol by an "e". Once one of the equivalent symbol bits has been declared in error, the entire symbol is in error. The system next corrupts the encoded codeword with the specified error pattern (an XOR operation) and displays the result as the "Received Codeword". This simulates the reception of a corrupted transmitted codeword by the decoder. The system (playing the part of a demodulator) forms a byte containing information pertaining to the symbols which have been specified (declared) erasures and hands this byte, along with the three byte received codeword (see Figure 2-6), to the decoder. It is at this point that the actual decoding algorithm begins and the timer is started. The decoder has no a priori knowledge of the selected message sequence or of the chosen error/erasure pattern. It merely takes the received codeword, calculates a table entry point from the message symbols, and starts the table search and compare operations at that point. In this case, since there are no errors or erasures, the table entry point is the correct codeword. Comparison with the received codeword shows no discrepancies, i.e., zero symbols are

unlike, and therefore the codetable word pointed to is declared the "Decoded Codeword". The remaining performance assessment information displayed indicates that, as a result of the decoding operation, the decoder found no symbol errors or erasures and found one codeword that was distance zero (i.e., an exact match) away from the received codeword. This codetable word was declared the "Decoded Codeword", output, and the timer stopped. The display shows a decoding "kernel" (table search and compare operation) time of 000.209 milliseconds or 209 microseconds. This time represents the minimum execution time required for one table fetch and compare operation. As such, it is the minimum time that the decoder takes to declare a decoded codeword when the codeword is received without errors or the corrupting error/erasure pattern affects only the parity check symbols. This time represents a limiting factor on the decoding rate (see 4.1.1).

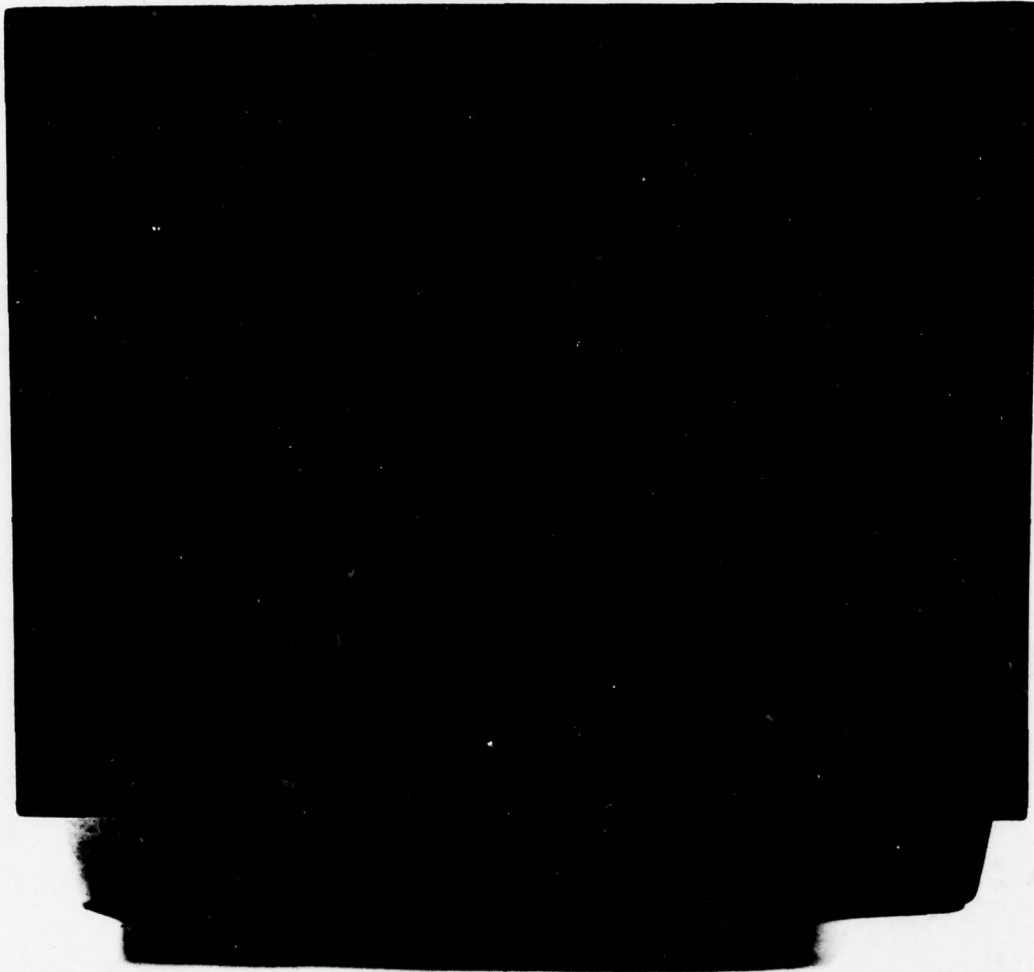


FIGURE 2-8: EXAMPLE 1-NO ERRORS AND NO ERASURES

2.3.2 Example 2 - Five Erasures

Figure 2-9 shows the results of decoding a received codeword that contains five erased symbols. Since the number of symbol erasures exceeds twice the maximum symbol error correcting capability of the code (equation (1)), this received codeword cannot be decoded and is output just as received (assuming soft decisions from the demodulator). The decoder performance assessment information shows an asterisk next to the "Decoder Codeword" indicating that the maximum number of allowable erasures has been exceeded. The decoding time was simply 4 microseconds, the time taken to recognize the excessive erasure condition and branch back to another received codeword. The number of codewords found is zero as would be expected. The number of symbol errors found and minimum distance (D) found are both equal to 8, a program initialization constant which is meaningless in this case. The number of symbol erasures found is five and accounts for the aborted decoding attempt.



FIGURE 2-9: EXAMPLE 2-5 ERASURES

2.3.3 Example 3 - Two Errors

Figure 2-10 shows the decoder performance assessment information for a received codeword which has been corrupted by an error pattern containing two symbol errors. Designating the right most symbol in the pattern as the 0th symbol, it can be seen that the operator has declared symbols 3 and 4 in error. Consequently, the received codeword differs from the encoded (transmitted) codeword only in symbols 3 and 4. Since two symbol errors, in the absence of any erased symbols, is an allowable (i.e., correctable) pattern from equation (1), a correct decoding operation should result. Indeed, as Figure 2-10 shows, the decoder did discover two symbol errors, found one word in the codetable that was distance two away and declared that table entry as the decoded codeword. It will be seen that a correct decoding occurred, since the declared "Decoded Codeword" is the same codeword as the originally transmitted "Encoded Codeword". The decoding time of 527 microseconds warrants further comment. Since it is known that a period of approximately 209 microseconds is required for each codeword table fetch and compare operation, it would seem that this particular received codeword required two table fetches. That this was indeed the case can be shown by recalling how the table entry point is calculated. From the received codeword, note that message symbols 6, 5, and 4 equal octal 152. Note that symbol 4 is in error and therefore the table entry point is not at the ideal location as in example 1. However, it is fortuitous that the error pattern changed symbol 4 from a value of 3 to 2, allowing the table entry point routine to direct the decoder to start searching the table at a position one codeword before the correct entry. Obviously, with other error patterns the entry point would be different resulting in greater decode times.

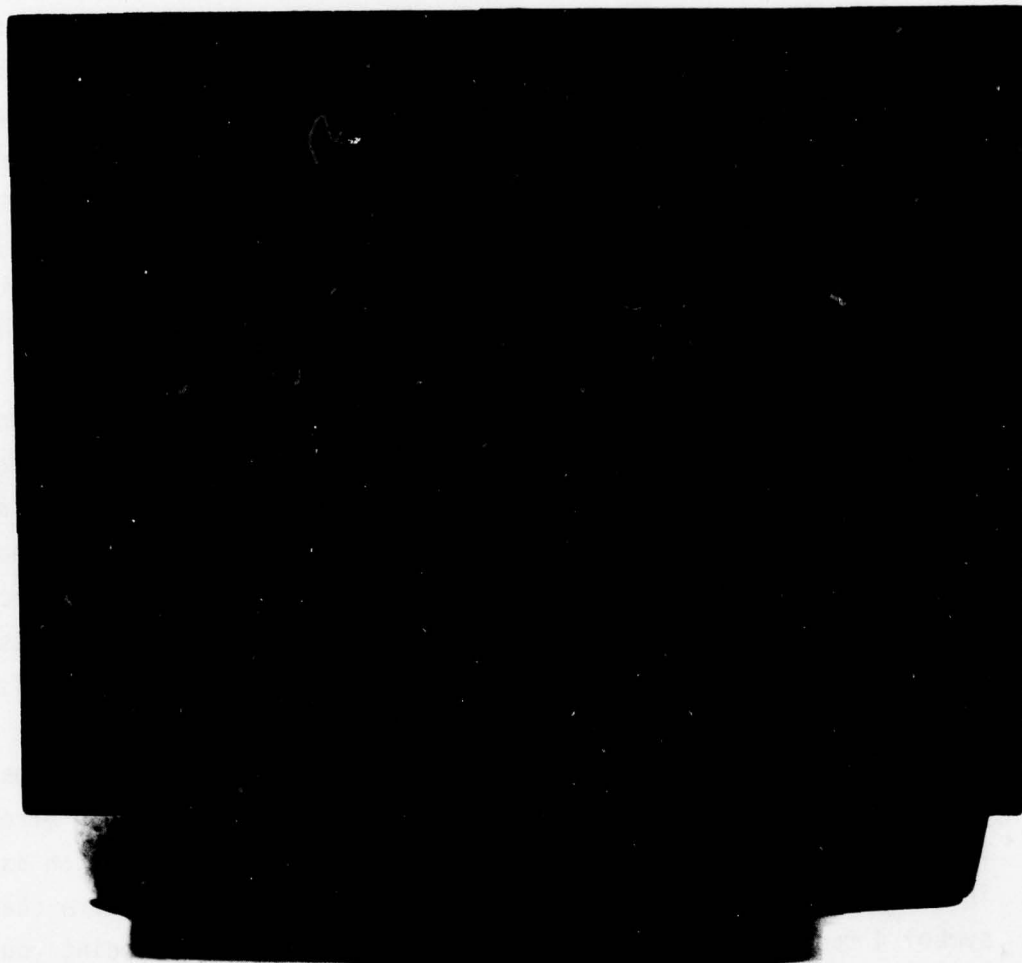


FIGURE 2-10: EXAMPLE 3-2 ERRORS

2.3.4 Example 4 - One Error and Two Erasures

Figure 2-11 shows the decoder performance assessment information for a received codeword which has been corrupted by an error and erasure pattern containing one symbol error and two symbol erasures. From equation (1) this is a correctable combination of errors and erasures, and a correct decoding operation should ensue. Referring to Figure 2-11, the decoder did discover one error and two erasures. The decoding time in this example was 8.505 milliseconds, indicating that about 40 out of the 512 codetable words were checked before finding the correct entry. Again, this time is dependent on how the error pattern affects the message symbols and therefore the calculated table entry point. The figure further indicates that the decoder found one codeword at a minimum distance of one symbol. This minimum distance value does not include the erased symbols declared in the received codeword, as these symbols are not counted in the symbol-by-symbol comparison checks. It is obvious that the declared "Decoded Codeword" is the same as the "Encoded Codeword" and therefore a correct decoding has occurred. It will be further noticed that the distance between the "Decoded Codeword" and the "Received Codeword" indeed is equal to one symbol, not counting the erased symbols.

It should be noted that erased message symbols are still utilized as received in the table entry point routine but not in the symbol compare operations. In this particular decoder implementation the value of the "erased" symbol is actually the correct value and thus the decoding times are favorably biased.

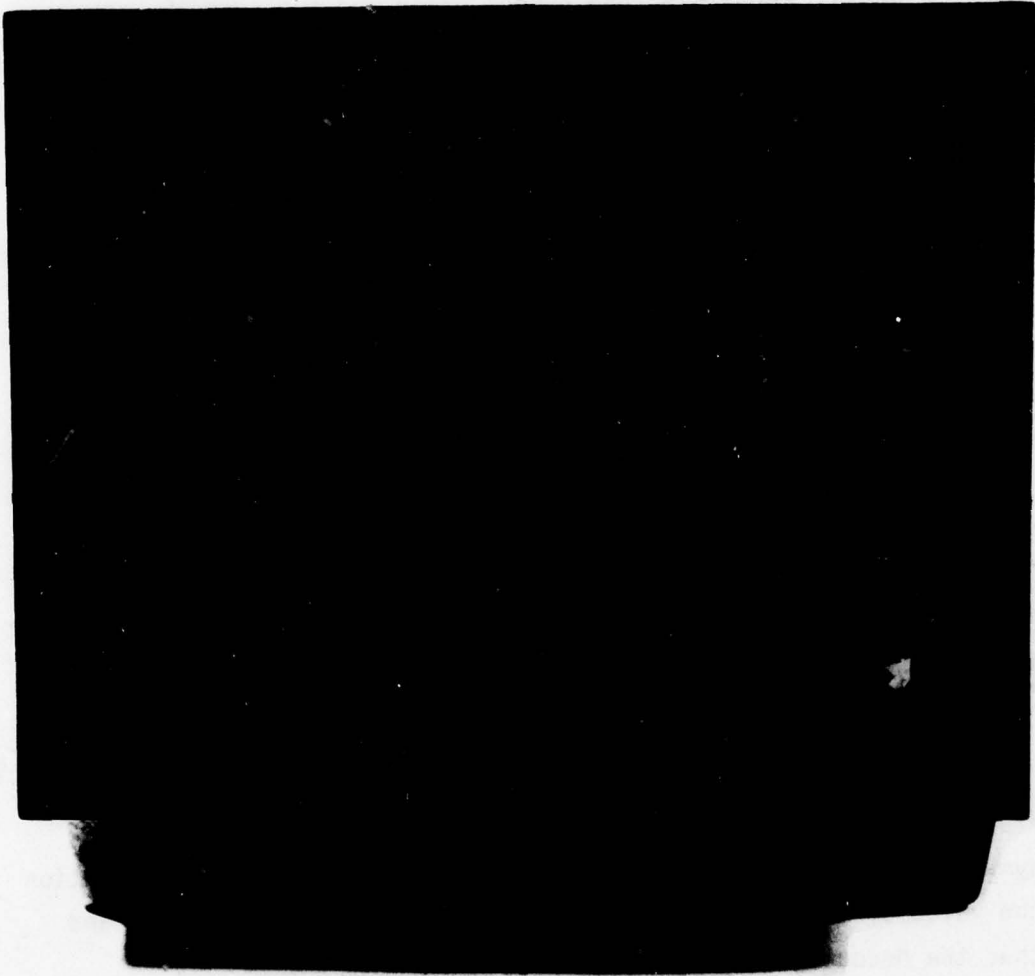


FIGURE 2-11: EXAMPLE 4-1 ERROR AND 2 ERASURES

2.3.5 Example 5 - Three Errors

Figure 2-12 shows the decoder performance assessment information for a received codeword which has been corrupted by an error pattern of three symbol errors. The value $t = 3$ ($s = 0$) does not satisfy equation (1) for the (7,3) R-S code, and therefore the error pattern exceeds the error correction capability of the code. The decoder will, however, attempt to decode the codeword since the number of declared erasures ($s = 0$) is within bounds. Figure 2-12 shows that the decoder did indeed recognize that three symbols were in error. However, when attempting to find a minimum distance codeword, the decoder searched the entire table (requiring 134.27 milliseconds) and found that there were seven words three symbols distant from the corrupted received codeword. All seven of these codewords were ambiguously declared as choices. Note that all seven of these codewords differ from the received codeword by only three symbols, and the originally encoded (transmitted) codeword is among them. Such a "list" of possible codewords can be viewed as a "partial decoding", i.e., reducing the uncertainty in codeword selection from one in 512 to one in seven. Given other a priori knowledge of the message source, such as its type, e.g., a slowly varying analog signal or redundant video data, might yield further clues as to the identity of the transmitted codeword.

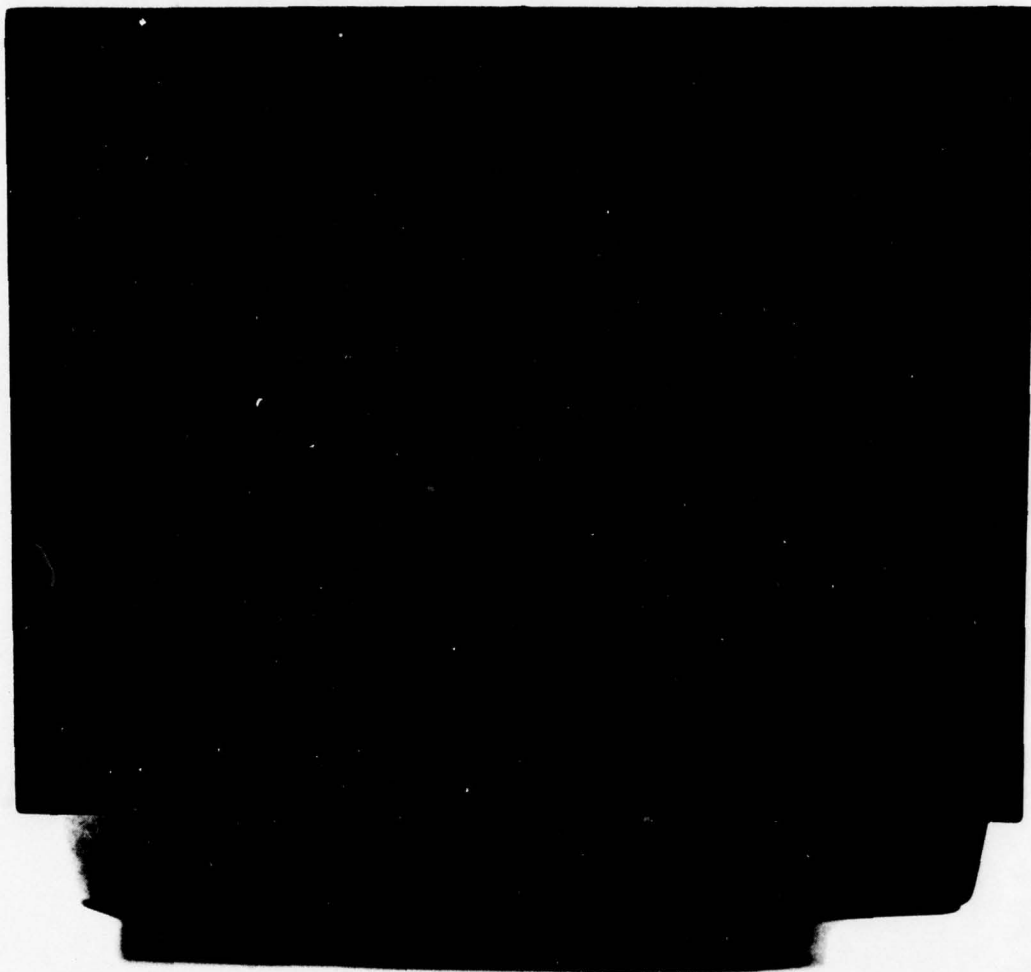


FIGURE 2-12: EXAMPLE 5-3 ERRORS

2.3.6 Example 6 - Three Errors and Two Erasures

Figure 2-13 shows the decoder performance assessment information for a received codeword which has been severely corrupted by a pattern of three errors and two erasures. Such a pattern clearly exceeds the correction-capability of the (7,3) R-S code, but the decoder attempts a decoding operation unaware of this fact. Indeed, as shown in Figure 2-13, the decoder reported that it successfully decoded this codeword and found one error and two erasures, taking 33.178 milliseconds to search approximately one quarter of the table before finding one codeword that was distance one away. Comparing the declared "Decoded Codeword" with the transmitted "Encoded Codeword", it can be readily seen that they do not agree and therefore a decoding error has occurred. What has happened, of course, is that the error/erasure pattern was so severe that it "mapped" (changed) the transmitted (encoded) codeword into a corrupted codeword that was closer to a different codeword than the original. In fact, comparing the declared "Decoded Codeword" with the "Received Codeword", it can be seen that (not counting erased symbols) only one symbol is not the same, i.e., the minimum distance between these two since the occurrence of this single error pattern is a much more "likely" event than the triple error pattern that actually occurred.



FIGURE 2-13: EXAMPLE 6-3 ERRORS AND 2 ERASURES

SECTION 3

DECODER TESTING

The operational parameters of the (7,3) R-S decoder were determined by a series of tests designed to exhaustively and randomly exercise the unit. The test objectives were to calculate the correctness of decoding, the average decoding rate, and the decoding delay.

3.1 Decoder Tests and Results

The interactive word-at-a-time operation described in Section 2.3 successfully decoded all submitted (correctable) error patterns. To gain further confidence in the correctness of decoding, exhaustive error pattern tests were run, whereby not only correct decoding was verified, but also average decoding times were calculated. To test the decoder under random error conditions, several Monte Carlo runs were completed, and average decoding times and correctness were again verified.

3.1.1 Exhaustive Tests

Each of the (7,3) R-S decoder's 512 valid codewords was subjected to every possible correctable error/erasure pattern shown in Table I. This "exhaustive" test ensured that the decoder could properly decode all possible combinations of the eight correctable error/erasure pattern types and also provided an opportunity to calculate the average decoding time for different error/erasure patterns. Table II shows the results of the exhaustive tests, with columns illustrating: (1) the correctable error/erasure pattern parameters, i.e., the number of erasures and/or errors per pattern, (2) the total number of patterns

TABLE II - EXHAUSTIVE TEST RESULTS

NUMBER OF SYMBOL ERRORS	NUMBER OF SYMBOL ERASURES	NUMBER OF PATTERNS RUN (512 x ALL POS- SIBLE COMBINATIONS)	AVERAGE PATTERN DECODING TIME
0	1	$\binom{7}{1} 512 = 3,584$	210 μ s
0	2	$\binom{7}{2} 512 = 10,752$	210 μ s
0	3	$\binom{7}{3} 512 = 17,920$	210 μ s
0	4	$\binom{7}{4} 512 = 17,920$	210 μ s
1	1	$\binom{7}{2} \cdot 2 \cdot 7 \cdot 512 = 150,528$	29.06 ms
1	2	$\binom{7}{3} \cdot 3 \cdot 7 \cdot 512 = 376,320$	29.06 ms
1	0	$\binom{7}{1} \cdot 7 \cdot 512 = 25,088$	29.06 ms
2	0	$\binom{7}{2} \cdot 7 \cdot 7 \cdot 512 = 526,848$	48.23 ms

2A-52,051

possible for each correctable case, and (3) the average decoding time for all members of that pattern's family. In addition, a partial run (time limited to three hours) was conducted for triple error patterns (an uncorrectable error pattern). The average list decoding time for this run was found to be 120.5 milliseconds or 89.7% of the search time for the complete table.

During the running of the exhaustive tests each declared "Decoded Codeword" was compared against the "Encoded (transmitted) Codeword" to verify correct decoding. In this manner one bit programmed in error in the codeword table ROMs was discovered and corrected. This error was easily spotted due to the consistent failure of the same codeword.

3.1.2 Monte Carlo Tests

In order to test the (7,3) R-S decoder under random error conditions, Monte Carlo runs were employed. The Monte Carlo runs were directed by a random number generator program written for the host system's CPU (MC6800). This random number generator program utilizes "Algorithm-M" and employs two random sub-generators: a linear congruential generator and a maximum length linear feedback shift register.^[3] The random number generator was used to select random codewords and random error/erasure patterns confined to the eight correctable types of Table I or II for each error/erasure pattern specified, three runs each of 32, 64, 128 and 256 random codewords and patterns were completed. As in the exhaustive tests, correct decoding and average decoding time were calculated for each run. Table III shows the results of the Monte Carlo runs with columns illustrating: (1) the error/erasure pattern specified, (2) the number of random runs allowed, and (3) the average decoding time per codeword over three runs.

TABLE III - MONTE CARLO TEST RESULTS

NUMBER OF SYMBOL ERRORS	NUMBER OF SYMBOL ERASURES	NUMBER OF RUNS	AVERAGE PATTERN DECODING TIME
0	1	32	210 μ s
0	1	64	209 μ s
0	1	128	210 μ s
0	1	256	210 μ s
0	2	32	210 μ s
0	2	64	209 μ s
0	2	128	209 μ s
0	2	256	210 μ s
0	3	32	209 μ s
0	3	64	210 μ s
0	3	128	209 μ s
0	3	256	210 μ s
0	4	32	209 μ s
0	4	64	210 μ s
0	4	128	209 μ s
0	4	256	210 μ s
1	1	32	30.6 ms
1	1	64	24.6 ms
1	1	128	27.5 ms
1	1	256	26.9 ms
1	2	32	27.9 ms
1	2	64	26.6 ms
1	2	128	27.6 ms
1	2	256	27.4 ms
1	0	32	26.6 ms
1	0	64	26.3 ms
1	0	128	28.1 ms
1	0	256	29.1 ms
2	0	32	35.2 ms
2	0	64	47.5 ms
2	0	128	46.7 ms
2	0	256	51.1 ms

14-52,850

3.2 TEST RESULTS ANALYSIS

The results are analyzed for both exhaustive and random tests to determine the correctness of the decoding operations and the average execution time for all correctable error/erasure patterns. For patterns beyond the design distance of the (7,3) R-S code, the decoder performance is analyzed with the aid of the classic standard array structure^[2].

3.2.1 Correctable Error/Erasure Patterns

After correcting a misprogrammed bit in the codeword table ROMs, the correctness tests ran without error for the exhaustive and Monte Carlo runs. Therefore, it was proven that the decoder will correctly decode all error/erasure patterns satisfying equation (1).

The average decoding times for each of the eight correctable error patterns warrant further analysis. Both tests (exhaustive and Monte Carlo) showed that the decoding time for any declared erasure pattern (in the absence of any errors) was approximately 209-210 microseconds. This is easily explained since the decoding algorithm ignores erased symbols and enters the table at the correct point the first time. Therefore, the average time shown is simply the time required for one table fetch and compare operation analogous to Example 1, Figure 2-8, for a correctly received word. It was mentioned for decoding Example 4 (2.3.4), however, that one critical programming liberty has been taken. This occurs when one or more of the codeword's information symbols has been declared erased. In 2.2.2 it was assumed that the soft decision erased symbol declarations were accompanied by "best guess" symbols. In calculating the table entry point using erased information symbols, however, the correct (encoded) symbol is always used. In other words, a routine was not included to "corrupt"

erased symbols like symbols which are in error. This should be considered for future incorporation and it most certainly affects average decoding times for patterns with erased information symbols.

The average decoding time for all patterns of one symbol error and 0, 1, or 2 erasures from the exhaustive tests was found to be 29.06 milliseconds. This number is subject to the remarks previously made relative to calculating table start addresses using erased symbol information. The random test runs for these same patterns provided average decode times that correlated very closely with the exhaustive averages. Figure 3-1 compares the results of the exhaustive vs. random tests of the average decoding time of the remaining four correctable error patterns. The total aggregate of random data points averages out to be only 5.6% below the decoding time of the exhaustive tests for the first three patterns, a figure highly dependent on the sample size of the random runs and the bias of the random number generator. For the two error and no erasure patterns, the small random sample run (i.e., $n = 32$) is seen to be below the exhaustive average by 27%. However, the random decoding times for this pattern correlate quite well with the exhaustive test results for larger sample runs, and, in fact, the aggregate of the four points averages only 6.4% below the average decoding time of the exhaustive tests.

3.2.2 Uncorrectable Error/Erasure Patterns

Rather extensive investigation was carried out relative to decoder performance for error/erasure patterns which do not satisfy equation (1) and lie therefore beyond the design distance of the (7,3) R-S code. Since the decoder decodes every possible received word into one (or more) possibly transmitted codewords, it exhibits a complete decoding algorithm and is not a bounded-distance-decoder in the strict sense^[4]. Specifically investigated were error-only patterns of weight three and

IA-52,847

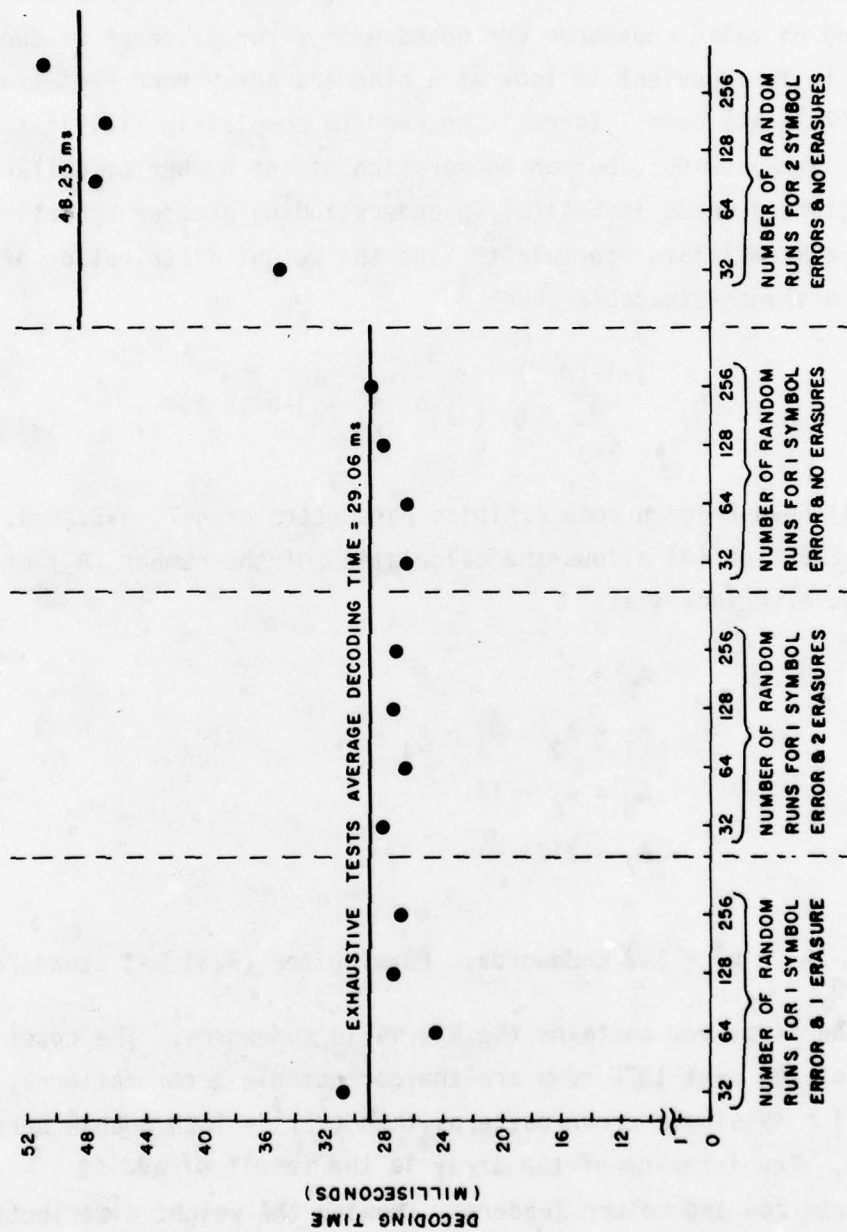


Figure 3-1 : EXHAUSTIVE VS. RANDOM TEST RESULTS

four. In order to better understand the decoder performance when operating on valid codewords corrupted with error patterns of such weight, it is expedient to look at a standard array representation of the (7,3) R-S code. There is no need to completely fill in the $2^n = 2^{21}$ array values, but an enumeration of the number of differently weighted words is helpful in understanding decoder operation. Using the MacWilliams' formula to find the weight distribution of a maximum-distance-separable code^[2]

$$A_j = \binom{n}{j} \sum_{h=0}^{j-1-(n-k)} (-1)^h \binom{j}{h} (q^{j-h-(n-k)} - 1) \quad (3)$$

the (7,3) Reed-Solomon code exhibits parameters of $n=7$, $q=8$, $k=3$, and $n-k=4$. Equation (3) allows the calculation of the number (A_j) of codewords of weight j as:

$$\begin{aligned} A_0 &= 1 \\ A_1 &= A_2 = A_3 = A_4 = 0 \\ A_5 &= A_6 = 147 \\ A_7 &= 217 \end{aligned}$$

where, $\sum_{j=0}^7 A_j = q^k = 512$ codewords. Forming the (7,3) R-S Standard

Array, the first row contains the 512 valid codewords. The coset leaders of the next 1078 rows are the correctable error patterns, first, $\binom{7}{1} = 49$ single error patterns, then $\binom{7}{2} 7^2 = 1029$ double error patterns. The interior of the array is the result of adding appropriate row and column leaders. Knowing the weight distribution of the valid codewords, and from R-S code theory that $d = n-k + 1$ and $t = 2$, it is seen that all single and double error patterns are

correctable. Thus the weight distribution of the interior of the array can be readily calculated since all single and double error patterns are present. For example, the number of words of weight six which are formed by combinations of valid codewords and the double error patterns (weight-two words) can be calculated by considering the various ways in which these combinations are possible. First, weight-6 words can be formed from weight-7 codewords added to a weight-2 error pattern where the mod-2 sum of one error symbol and one codeword symbol equals zero, and the sum of the other error symbol and a codeword symbol results in a different non-zero symbol. Likewise, a weight-6 word will result from a six-weight codeword where two non-zero symbols are changed by the two error pattern symbols into two other non-zero symbols. Furthermore, a six-weight codeword can be combined with a two-weight error such that one non-zero error symbol value adds to a zero symbol of the codeword while the other non-zero error symbol value exactly equals the corresponding symbol of the codeword and yields a zero symbol. Lastly, a five-weight codeword can combine with a two-weight error pattern whereby one non-zero codeword symbol is added to one non-zero error symbol resulting in a different symbol, and one zero symbol of the codeword is added to the other non-zero symbol of the error pattern resulting in a new non-zero symbol. Mathematically, these combinations can be calculated as:

$$W_{62} = \binom{7}{1}\binom{6}{1} 6 \cdot 217 + \binom{6}{2} 6^2 \cdot 147 + 7 \cdot 6 \cdot 147 + 7\binom{5}{1} \cdot 6 \cdot 2 \cdot 147 = 201,978 \quad (4)$$

Figure 3-2 illustrates the structure of the standard array for the (7,3) R-S code with the calculated array weight enumerators included for the correctable error patterns only. Considering for a

IA-52,857

$q^k = 512$

CODEWORDS	
C_0	C_1, C_2, \dots
SINGLE ERROR PATTERNS $W_1 = 49$	$(W_5 = 147, W_6 = 147, W_7 = 217)$ $\left. \begin{array}{l} W_4 = 735 \\ W_5 = 5292 \\ W_6 = 8869 \\ W_7 = 35182 \end{array} \right\} = 25,039 = 511 \times 49$
DOUBLE ERROR PATTERNS $W_2 = 1029$	$\left. \begin{array}{l} W_3 = 1470 \\ W_4 = 19845 \\ W_5 = 94227 \\ W_6 = 201978 \\ W_7 = 208299 \end{array} \right\} = 525,819 = 511 \times 1029$
ERROR PATTERNS OF WEIGHT ≥ 3 (3017)	1,541,687 (WEIGHT DISTRIBUTION NOT CALCULATED)

$q^k = 4096$

Figure 3-2: (7,3) R-S STANDARD ARRAY

moment array members of weight three, the figure shows that 1470 of these result from combinations of two-weight error patterns and valid codewords of

$$W_3 = E_2 + C_i \quad (5)$$

where: W_3 = interior array words of weight three

E_2 = double error pattern

C_i = any i^{th} codeword

If one of these three-weight words is an error pattern that corrupts a transmitted valid codeword C_j , then:

$$C_j + W_3 = C_j + E_2 + C_i = E_2 + C_k \quad (6)$$

where $C_i + C_j = C_k$ by the group property of the linear block code.

Equation (6) indicates that 1470 weight-3 error patterns will "map" any transmitted codeword closer (distance 2) to another valid codeword. What of the remaining three-weight error patterns? There is a total of $\binom{7}{3} 7^3 = 12,005$ weight-3 error patterns. Since it has just been shown that 1470 of these result in incorrect decodings, the remaining 10,535 triple error patterns must result in corrupted codewords that are at least distance three away from one or more codewords. Figure 3-2 shows that a maximum of 3017 of these patterns can be arbitrarily declared coset leaders, and therefore a "standard array decoder" could ambiguously decode 3017 out of the 12,005 or 25% of all triple error patterns. The (7,3) R-S decoder algorithm under consideration here, will "decode" 10,535 corrupted codewords, 87.8% of the total triple error patterns, to a list of possibly transmitted codewords. Extensive decoding runs have confirmed that this list size is always ≤ 7 .

Considering four-weight (quad) error patterns, further runs showed that 490 out of the 84,035 possible four-weight error patterns (0.6%) corrupting any transmitted codeword result in a decoded list of 35 possibly transmitted codewords. 62,965 (75% of all) quad error patterns map any transmitted codeword distance 3 away from 7 or fewer valid codewords (all incorrect). As shown in Figure 3-2, 19,845 (24% of all) quad error patterns map any transmitted codeword distance 1 away from 1 valid codeword. These last two cases appear as "proper decodes" from the decoder's point of view, but are in fact incorrect.

SECTION 4

CONCLUSIONS

The design, construction, and testing of the microprocessor-based (7,3) R-S decoder resulted in many interesting conclusions. Analysis of the decoder's performance enables simple conclusions to be made relative to the decoding rates obtainable and the decoding delay. Conclusions are also drawn relative to system lessons learned, testing error coding systems, and improving decoder performance.

4.1 SYSTEM IMPLICATIONS

If this simple (7,3) R-S decoder were to be used as a design resource in the engineering of a communications channel, its operation would certainly affect the reliability and performance of that channel. System design aspects, such as buffering requirements, depend to a large extent on the decoding rate and delay of the decoder, while decoding rates obtainable affect channel rates and consequently throughput.

4.1.1 Average Decoding Rate

Since the decoding time per codeword is dependent on the particular error/erasure pattern that corrupts the word, the decoding rate will vary. Figure 4-1 shows the average decoding rate of the (7,3) R-S decoder vs. symbol error rate. The abscissa of the graph is defined in both information symbols per second and information bits per second, the latter being three times the former since one eight-level symbol is the equivalent of three bits of information. Assuming that the (7,3) R-S decoder is matched to an m-ary (8-level) channel, the ordinate relates the random error rate on the channel relative to the independent transmission of symbols. The dashed line on the graph is

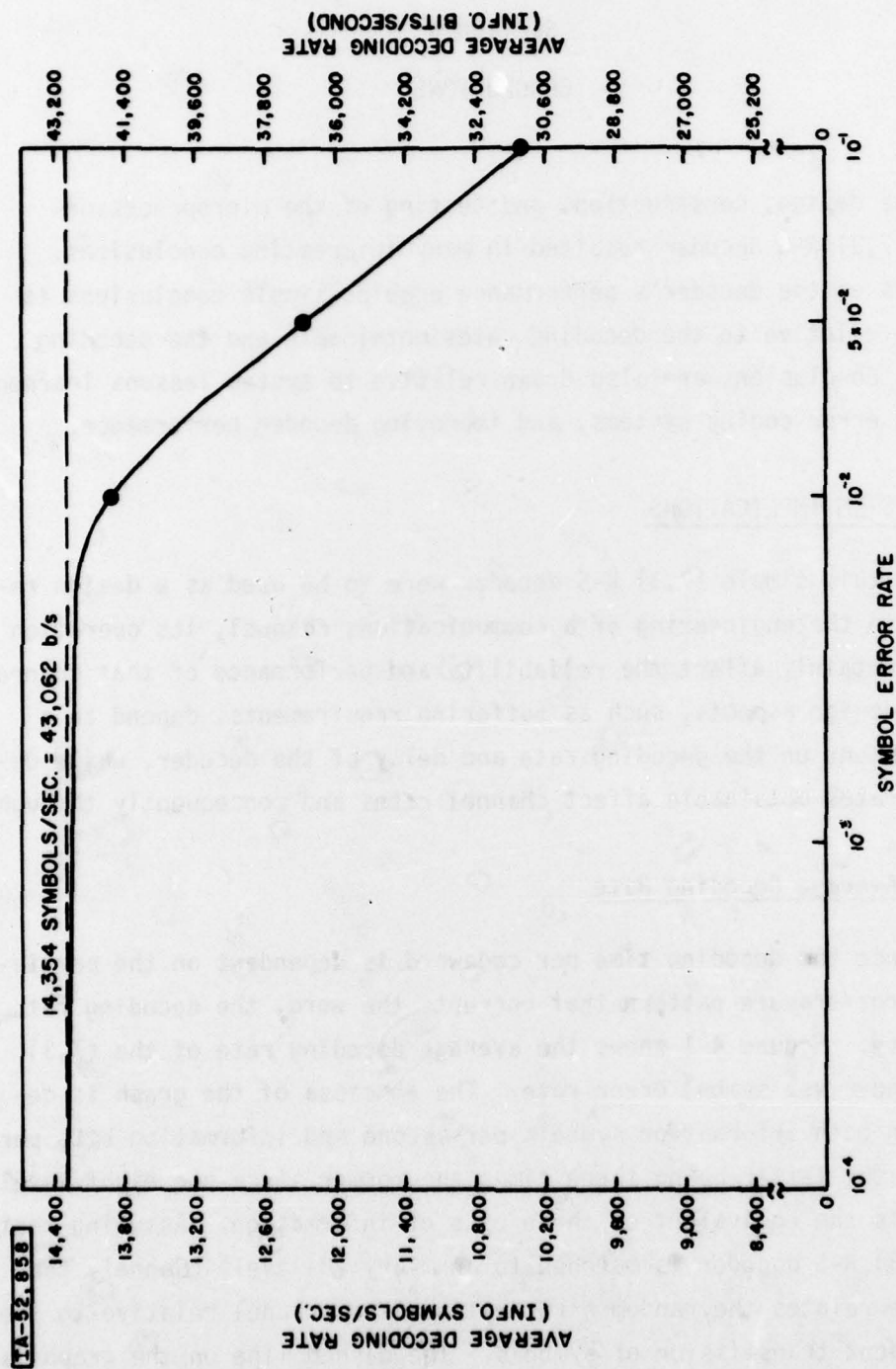


Figure 4-1: DECODING RATE vs SYMBOL ERROR RATE

an asymptote of 14,354 symbol/sec. In 2.3.1 and 3.2.1 it was shown that 209 microseconds is the minimum decoding time obtained when only one table fetch and compare operation is necessary. Therefore, the reciprocal of this time (i.e., $1/209 \times 10^{-6}$) represents the maximum decoding rate obtainable in all codewords were received correctly or if correctable error patterns are confined to check symbol positions only. The curve relating the average decoding rate for a given symbol error rate was calculated via equation (7):

$$R_{(SER)} \approx \frac{P_{oe} + P_{1E,c} + P_{2E,c}}{209 \times 10^{-6}} + \frac{P_{1E,m} + P_{2E,m}}{67 \times 10^{-3}} + \frac{P_{3E}}{134 \times 10^{-3}} \quad (7)$$

where $R_{(SER)}$ = average decoding rate at some specific symbol error rate (SER)

P_{iE} = the probability of receiving a codeword with i errors at the specified SER
 $= \binom{7}{i} SER^i (1-SER)^{7-i}$

$P_{iE,c}$ = the probability of receiving a codeword with i errors, all of which are confined to the check symbol positions

$P_{iE,m}$ = the probability of receiving a codeword with i errors, one or more of which are located at a message symbol location

As the curve shows for SERs better than approximately 10^{-2} most symbols are received correctly and the rate asymptotically approaches the correct codeword recognition limit. It must be remembered that these rates were obtained using a 1 MHz CPU chip. A 2 MHz 6800B CPU chip would result in a doubling of these rates.

4.1.2 Decoder Delay

Like the instantaneous decoding rate, the delay inherent in the operation of the (7,3) R-S decoder varies with the error/erasure pattern. Obviously, the longest decoding delay results from a complete codeword table search and is approximately 134 milliseconds. The probability that a correctable error pattern of one or two errors will corrupt message symbols (hence the table entry point) is approximately (from equation 7)

$$P \approx P_{1E,m} + P_{2E,m} \quad (8)$$

Assuming that this corruption always results in an entire table search, equation (8) is a loose upper bound on the probability of this event. For example the likelihood of this event for symbol error rates below 10^{-2} is less than .029 due to correctable error patterns, and it is less than .000034 for uncorrectable error patterns.

Good system design dictates the incorporation of a codeword buffer. The width of the codeword buffer will, in general, be seven symbols or 21 bits, while the depth is dependent on decoder delay and channel rate. Choosing a particular channel transmission rate, the channel's information-transfer efficiency, i.e., the throughput, can be calculated^[5]. Many factors influence a channel's throughput rate, such as the channel's transmission rate, the amount of overhead symbols dedicated to control synchronization and coding redundancy, channel error rate, and block length. Ideally, the decoding rates of Figure 4.1 for known channel error conditions could be used to set the channel transmission rate. Consequently, enough buffer depth must then be designed to hold received codewords during intervals of decoding that are longer than the reciprocal of the instantaneous transmission (decoding) rate. Without resorting to a detailed queueing

analysis, a heuristically engineered buffer design can be proposed. For example, at a symbol error rate of 10^{-2} , the average decoding rate shown in Figure 4-1 is approximately 13,800 information symbols per second (i.e., 4600 codewords per second or 41.4 Kb/s). If this is set as the channel transmission rate, then the buffer must be large enough to hold received codewords during the longest decode time - 135 milliseconds for a complete table search. This establishes a buffer length of $(134 \times 10^{-3})(4600) = 616$ codewords of 21 bits/codeword, or 1.6 K bytes. The probability of a complete table search (P_{cts}) from (8) is .029. The probability of buffer overflow due to two complete table searches in a sequence of 616 codewords is $\binom{616}{2} P_{cts}^2 (1-P_{cts})^{164} = 2.264 \times 10^{-6}$. Conservatively designing a buffer twice this estimate size or 3.2K bytes would provide an extremely low probability of over/under-flow and would at the same time not become a hardware burden. Where hardware constraints are critical and information rate is not, setting the transmission rate lower than the decoding rate provides the opportunity for a tradeoff between buffer size and throughput as well as extended performance with larger codes and/or list decoding techniques.

4.2 LESSONS LEARNED

From the construction and testing of the simple (7,3) R-S decoder a number of lessons learned and conclusions can be related:

1. 8-bit MOS microprocessors can do error coding/decoding operations at reasonable decoding rates with minimum hardware if small block codes are utilized. The key to utilizing low-cost 8-bit MOS microprocessors for error coding lies in restricting the chosen code to a

short block length, and using simple decoding algorithms with table-look up procedures wherever possible. As future generations of microprocessors become more powerful, these restrictions can undoubtedly be relaxed.

2. In representing q-ary code symbols in binary-based microprocessor designs, codeword representation is critical in order to minimize decoding time. The binary representation of q-ary code symbols must be designed to make maximum use of the microprocessor's instruction set.
3. Simple decoding algorithms will achieve minimum execution time if in-line code (i.e., no subroutine calls, loops, etc.) is extensively used in critical portions of the program. This results in a tradeoff between memory storage requirements and execution speed, a viable alternative given the current trend of increasing memory densities and lower per/bit costs of semiconductor RAM/ROM.
4. Using a maximum-likelihood decoding algorithm with table look-up, smart table search procedures must be implemented to speed table search times. Using certain symbols of systematic or nonsystematic codes, table entry points can be calculated that result in highly possible minimum table search time.
5. Having decoded to a valid codeword, faster message extraction is facilitated by the use of systematic codewords obviating the necessity of a final polynomial divide operation. Alternatively, when using

non-systematic codes memory storage for the corresponding message symbols can be traded off against the added execution time of a polynomial divide operation.

6. The instruction set of a microprocessor establishes the capability of that CPU to efficiently perform a given decoding algorithm. The lack of or inclusion of certain instructions may dictate an entirely different coding approach for the algorithm resulting in radically different execution times and memory storage requirements.
7. The amount of data and program storage space necessary to implement the (7,3) R-S decoder in a multi-chip microprocessor configuration easily lends itself to an economical design utilizing some of the state-of-the-art one chip microcomputers such as the MOSTEK 3870.

4.3 ERROR CODING SYSTEM TESTING

Experience gained in testing the microprocessor-based (7,3) R-S decoder has provided the basis for a general test methodology for error coding systems. Whereas exhaustive error/erasure pattern tests on the (7,3) decoder were feasible due to the short block lengths, and the small number of bits per symbol, such tests would be prohibitively long running on larger codes. In fact, even for the (7,3) R-S decoder error/erasure patterns larger than three symbols were not run due to the size of the combination set (e.g., 823,543 for 6-symbol) error/erasure patterns). It may, however, in many instances be feasible to exhaustively run all guaranteed correctable error/erasure patterns. From

such tests correct decoding could be verified for a bounded distance decoder, and statistical data on decoding times could be gathered. For longer codes and more sophisticated decoding methods, some sort of Monte Carlo random simulation testing makes the most sense. With a statistically verifiable Monte Carlo exercising program, decoder performance could be quickly assessed under varying message source and error environment conditions. Complete decoder evaluation is difficult without knowledge of the particulars of the system in which it will be employed. Lacking this information, assumptions can be made relative to desired channel transmission rates, error conditions, synchronization and control overhead schemes, to enable the calculation of total system throughput. From these preliminary findings the assumed parameters, as well as the error coding parameters, can be varied to achieve the desired system performance. Decoder test data is invaluable in this respect ensuring a certain level of performance from the stand-alone decoder.

4.4 (7,3) R-S DECODER IMPROVEMENTS

While analyzing the test results of the (7,3) R-S decoder, it becomes apparent that one design assumption led to a slight biasing of the decode times. As reported in 3.2.1, this event arises from codeword table entry point calculations using erased symbols. It was stated that a soft-decision demodulator would pass code symbol estimates to the decoder along with a confidence measure, which in the simplest case is an erased/not-erased indicator. The (7,3) R-S decoder as tested, however, does not quite approximate this operation. The problem comes about from the fact that the operator- (or random generator) entered codeword erased symbols are not corrupted, but are passed unchanged to the table entry point routine. This biases the decoding times, since the correct table entry usually results. The

decoder operation should be changed to corrupt these erased symbols for a more accurate simulation. Having done this, the danger then exists that if the "best guess" erased symbol is a message symbol it may cause the table entry point routine to start table search "beyond" the correct codeword resulting in abnormally long table searches. Fortunately, there is an easy "algorithm fix" for this eventuality. If a declared erased symbol is the least significant or next to least significant message symbol (symbols 4 and 5, Figure 2-5), the codeword table entry point routine shall replace the best-guess value of these symbols with zero. This will ensure that the table entry point is not beyond the correct codeword location if the most significant message symbol is correct. Setting symbol 4 = 0 results in a worse case search of 8 codewords. Likewise, symbol 5 = 0 may result in a search of up to 64 codewords. Unfortunately, this fix will not result in any average time savings if applied to message symbol 6. This "fix" is, in essence, making use of the erasure symbol information (i.e., the location of a possible symbol error is known). Using this information, the algorithm can ensure that maximum table searches will not occur for error patterns satisfying the three following conditions:

1. A correctable error/erasure pattern is involved.
2. Message symbol 6 is not erased nor in error.
3. Message symbol 5 and/or 4 may be erased.

Additionally, this procedure may be employed dependent on other soft-decision information from the demodulator (e.g., received S/N ratio) indicating the degree of confidence in the declared symbol.

Recent work on simple (7,3) R-S decoder implementations done by others on Project 7010 and documented elsewhere has resulted in

considerable improvement in decoder operation. By using multiple codetables, minimum symbol storage, and simple finite field arithmetic operations, decoder performance can be dramatically improved over the results reported on here.

LIST OF REFERENCES

1. Skoog, E. N., "Error Correction Coding with NMOS Microprocessors, Concepts", ESD-TR-79-125, Vol. I, Electronic Systems Division, AFSC, Hanscom AFB, MA, May 1979.
2. Peterson, W. W., and Welson, Jr., E. J., Error-Correcting Codes, The MIT Press, 1972, pp. 52-56.
3. Knuth, D. E., The Art of Computer Programming, Volume 2, Semi-numerical Algorithms, John Wiley, 1975, pp. 20-32.
4. Berlekamp, E. R., Algebraic Coding Theory, McGraw-Hill, 1968, pp. 2.
5. Boustead, C. N. and Mehta, K., "Getting Peak Performance on a Data Channel", Data Communications, Volume 3, No. 2, July/August 1974, pp. 39-47.